IN THE CLAIMS

1. (Original) A method for circuit emulation over a multi-packet label switching (MPLS) network, comprising the steps of:

receiving a time division multiplexed data stream at an ingress end;

dividing said data stream into a set of fixed sized packets;

adding a service header to each of said packets;

adding an additional header on top of said service header in accordance with MPLS protocols;

removing said additional header after each packet has been processed by said MPLS network; and

using said service header to recover said data stream at an egress end.

2. (Original) The method of claim 1, further comprising the steps of:

monitoring said data stream; and

attaching an alarm bit in a service header of a subsequent packet if a break in said data stream is detected.

3. (Currently Amended) The method of claim 1, further comprising the ~~steps~~ step of:

using a structure pointer in said service header to indicate whether a header byte in a synchronous payload envelope is present within a packet, said structure pointer indicating the location of said header byte in said packet.

4. (Original) The method of claim 3, further comprising the step of:

reserving a pointer value indicating that said header byte is not present within said packet.

5. (Original) The method of claim 1, further comprising the steps of:

recording a stuffing time difference in a service header at said ingress end; and

implementing said stuffing time difference at said egress end.

6. (Original) The method of claim 1, further comprising the steps of:

(a) storing a first set of frames into a data buffer;

(b) calculating a first data average of said first set of frames in said data buffer to obtain a threshold value;

(c) storing a next set of frames into said data buffer;

(d) calculating a next data average of said next set of frames in said data buffer;

(e) comparing said next data average to said threshold value;

(f) if said next data average is greater than said threshold value:

(1) generating a negative justification indicator; and

(2) sending one more byte at said egress end;

(g) if said next data average is less than said threshold value:

(1) generating a positive justification indicator; and

(2) sending one less byte at said egress end; and

(h) repeating said steps (c )-(g).

7. (Original) The method of claim 1, further comprising the steps of:

checking a sequence counter in said service header of each packet in said set of packets;

locating at least one header byte in said set of packets;

measuring all bytes between two header bytes; and

pushing said set of packets into a frame.

8. (Original) The method of claim 1, further comprising the steps of:

checking a sequence counter in said service header of each packet in said set of packets to determine if all packets are present sequentially; and

inserting a dummy packet if a packet is missing in said set of packets.

9. (Original) The method of claim 8, further comprising the steps of:

receiving an out of sequence packet; and

discarding said out of sequence packet.

10. (Original) The method of claim 1, further comprising the steps of:

checking a sequence counter in said service header of each packet in said set of packets to determine if all packets are present sequentially;

terminating a current connection if multiple packets are missing in said set of packets;

discarding said set of packets; and

establishing a new connection to begin receiving packets.

11. (Original) The method of claim 1, further comprising the steps of:

checking a sequence counter in said service header of each packet in said set of packets to determine if all packets are present sequentially; and

establishing an in-frame condition after said set of packets are received in sequence.

12. (Original) The method of claim 11, further comprising the steps of:

determining whether said in-frame condition is valid; and

terminating a current connection if said in-frame condition is not valid.

13. (Original) A computer program product for circuit emulation over a multi-packet label switching (MPLS) network, comprising:

logic code for receiving a time division multiplexed data stream at an ingress end;

logic code for dividing said data stream into a set of fixed sized packets;

logic code for adding a service header to each of said packets;

logic code for adding an additional header on top of said service header in accordance with MPLS protocols;

logic code for removing said additional header after each packet has been processed by said MPLS network; and

logic code for using said service header to recover said data stream at an egress end.

14. (Original) The computer program product of claim 13, further comprising:

logic code for monitoring said data stream; and

logic code for attaching an alarm bit in a service header of a subsequent packet if a break in said data stream is detected.

15. (Original) The computer program product of claim 13, further comprising:

logic code for using a structure pointer in said service header to indicate whether a header byte in a synchronous payload envelope is present within a packet, said structure pointer indicating the location of said header byte in said packet.

16. (Original) The computer program product of claim 15, further comprising:

logic code for reserving a pointer value indicating that said header byte is not present within said packet.

17. (Original) The computer program product of claim 13, further comprising:

logic code for recording a stuffing time difference in a service header at said ingress end; and

logic code for implementing said stuffing time difference at said egress end.

18. (Original) The computer program product of claim 13, further comprising:

(a) logic code for storing a first set of frames into a data buffer;

(b) logic code for calculating a first data average of said first set of frames in said data buffer to obtain a threshold value;

(c) logic code for storing a next set of frames into said data buffer;

(d) logic code for calculating a next data average of said next set of frames in said data buffer;

(e) logic code for comparing said next data average to said threshold value;

(f) if said next data average is greater than said threshold value:

(1) logic code for generating a negative justification indicator; and

(2) logic code for sending one more byte at said egress end;

(g) if said next data average is less than said threshold value:

(1) logic code for generating a positive justification indicator; and

(2) logic code for sending one less byte at said egress end; and

(h) logic code for repeating said (c)-(g).

19. (Original) The computer program product of claim 13, further comprising:

logic code for checking a sequence counter in said service header of each packet in said set of packets;

logic code for locating at least one header byte in said set of packets;

logic code for measuring all bytes between two header bytes; and

logic code for pushing said set of packets into a frame.

20. (Original) The computer program product of claim 13, further comprising:

logic code for checking a sequence counter in said service header of each packet in said set of packets to determine if all packets are present sequentially; and

logic code for inserting a dummy packet if a packet is missing in said set of packets.

21. (Original) The computer program product of claim 20, further comprising:

logic code for receiving an out of sequence packet; and

logic code for discarding said out of sequence packet.

22. (Original) The computer program product of claim 13, further comprising:

logic code for checking a sequence counter in said service header of each packet in said set of packets to determine if all packets are present sequentially;

logic code for establishing an in-frame condition after all packets for a frame are received in sequence;

logic code for terminating a current connection if multiple packets are missing in said set of packets;

logic code for discarding said set of packets; and

logic code for establishing a new connection to begin receiving packets.

23. (Original) The computer program product of claim 22, further comprising:

logic code for checking a sequence counter in said service header of each packet in said set of packets to determine if all packets are present sequentially; and

logic code for establishing an in-frame condition after the set of packets are received in sequence.

24. (Original) The computer program product of claim 23, further comprising:

logic code for determining whether said in-frame condition is valid; and

logic code for terminating a current connection if said in-frame condition is not valid.